



LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

An Efficient Approach for Solving Large Stochastic Unit Commitment Problems Arising in a California ISO Planning Model

T. Parriani, G. Cong, C. Meyers, D. Rajan

December 4, 2013

IEEE Power and Energy Society General Meeting 2014
National Harbor, MD, United States
July 27, 2014 through July 31, 2014

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

An Efficient Approach for Solving Large Stochastic Unit Commitment Problems Arising in a California ISO Planning Model

Tiziano Parriani
University of Bologna
Bologna, Italy

Guojing Cong
IBM TJ Watson Research Center
Yorktown Heights, NY USA

Carol Meyers and Deepak Rajan
Lawrence Livermore National Laboratory
Livermore, CA USA

Abstract—We describe our experience in obtaining significant computational improvements in the solution of large stochastic unit commitment problems. The model we use is a stochastic version of a planning model used by the California Independent System Operator, covering the entire WECC western regional grid. We solve daily hour-timestep stochastic unit commitment problems using a new progressive hedging approach that features linear subproblems and guided solves for finding feasible solutions. For stochastic problems with 5 scenarios, the algorithm produces near-optimal solutions with a 6 times improvement in serial solution time, and over 20 times improvement when run in parallel; for previously unsolvable stochastic problems, we obtain near-optimal solutions within a couple of hours. We note that although this algorithm is demonstrated for stochastic unit commitment problems, the algorithm itself is suitable for application to generic stochastic optimization problems.

Index terms— integer linear programming, optimization methods, parallel algorithms, power generation planning

I. INTRODUCTION

The state of California has embarked on an aggressive plan to produce 33% of its electric energy from renewable resources by the year 2020 [1]. The increased penetration of intermittent renewable generation needed to meet this goal will substantially increase the variability and uncertainty in generation resources available to system operators. To assess the impact of such high renewable penetrations, the California Energy Commission funded a recently completed study at Lawrence Livermore National Laboratory to couple atmospheric models capable of producing renewable generation trajectories with a stochastic day-ahead unit commitment optimization model [2]. This stochastic day-ahead unit commitment model employs at its core a deterministic unit commitment planning model developed by the California Independent System Operator (ISO) for their study of market impacts under the 33% renewable portfolio standard [3]. The deterministic model is based on a grid description and operational specifications provided by the California ISO, and is implemented using the PLEXOS power market software package [4], which generates a mixed-integer linear programming formulation suitable for determining day-ahead hourly unit commitment decisions.

The stochastic unit commitment model is formulated as a two-stage mixed-integer stochastic optimization extension of the deterministic model, where scenarios are defined by

different renewable generation trajectories, unit commitment states for long-start generators are treated as first-stage decisions (common across all scenarios), and economic dispatch values and unit commitment states for short-start generators are treated as second-stage decisions (one for each scenario).

A. Model Description and Prior Computational Performance

The model representation of the Western Energy Coordinating Council (WECC) grid developed by the California ISO includes more than 2,400 generating units, over 42 zones in 11 states, with 120 transmission lines between zones (a zonal model). Wind and solar (PV) inputs are included at a zonal level, aggregated from numerous individual real and proposed wind and solar sites. The California ISO model calculates hourly day-ahead unit commitments for all 2,400 generating units, with integer commitments for all (several hundred) generation units in California, and fractional commitments elsewhere. As a result, the deterministic day-ahead mixed-integer program (MIP) is already fairly large, including roughly 400,000 constraints, 600,000 continuous variables, 10,000 general integer variables and 2,000 binary variables.

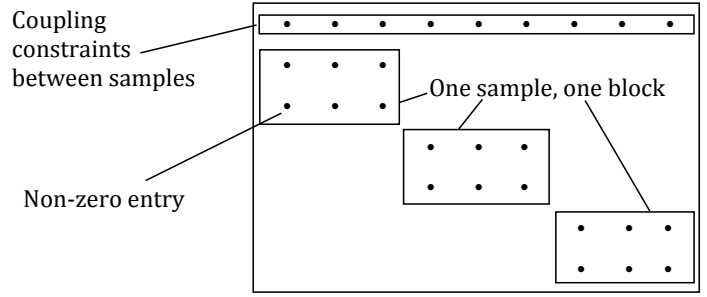
Variables and constraints in the stochastic version of the model are (roughly) linear multiples of the corresponding values in the deterministic model, scaling with the number of scenarios used. Thus, the computational burden associated with solving such problems becomes prohibitive for even very powerful systems. In the California Energy Commission study [2], it was found that for only 8 scenarios, each day-ahead stochastic unit commitment problem already required an average of 5 hours to solve, and no solutions at all were found for 20 or more scenarios. It was for this reason the original study was downscaled to include only 5 scenarios [2].

B. Contributions

We propose in Section III a new method for solving two-stage stochastic optimization problems that significantly reduces the solution time of the aforementioned stochastic unit commitment problems. As detailed in Section IV, this method allows us to solve individual unit commitment problems roughly 6 times faster than previously achievable, while a parallel implementation of this method (many of its stages are parallelizable) solves the problems more than 20 times faster.

$$\begin{aligned}
& \min \sum_s p_s (c^T x_s + q_s^T y_s) \\
& \text{subject to:} \quad x_1 = x_2 = \dots = x_S \quad (1a) \\
& \quad \quad \quad Ax_s = b \quad (1b) \\
& \quad \quad \quad T_s x_s + W_s y_s = h_s \quad (1c) \\
& \quad \quad \quad l_1 \leq x_s \leq u_1, \quad l_2 \leq y_s \leq u_2 \\
& \quad \quad \quad x_s^j \in \mathbb{Z}, \quad \forall j \in D_1 \subseteq \{1, \dots, n_1\} \\
& \quad \quad \quad y_s^j \in \mathbb{Z}, \quad \forall j \in D_2 \subseteq \{1, \dots, n_2\}
\end{aligned}$$

(a) Mathematical Formulation: Constraints (1b) and (1c) model the first- and second-stage decisions, respectively, and (1a) ensure that first-stage decisions are identical across scenarios.



(b) Pictorial Representation of non-zero entries in constraint matrix: Coupling constraints (1a) have first-stage variables from all scenarios. Constraints (1b) and (1c) can be divided into blocks, one for each scenario s .

Figure 1. The deterministic equivalent formulation of the stochastic problem has a specific block-angular structure that allows for dual decomposable schemes.

II. DUAL DECOMPOSABLE SCHEMES FOR STOCHASTIC OPTIMIZATION PROBLEMS

We begin by reviewing stochastic optimization problems, and briefly describe schemes for solving such problems. We then consider the Progressive Hedging (PH) algorithm in detail, highlighting many of its strengths and some weaknesses.

A. Stochastic Optimization: An overview

A stochastic optimization problem is often formulated as a two-stage optimization problem. Given a set of scenarios $\omega_s, s \in [1, S]$ with corresponding probabilities p_s , the sample-based MIP is formulated as in Figure 1(a), where x and y are first-stage and second-stage decision variables, respectively.

This formulation is often referred to as dual decomposable, since by eliminating the coupling constraints (1a) the problem can be decomposed into a separate subproblem for each scenario; see Figure 1(b). Many dual decomposition approaches have been applied to such problems, including Lagrangian [5], and augmented Lagrangian methods [6], branch and price approaches based on Dantzig-Wolfe decomposition [7], and the Progressive Hedging (PH) algorithm proposed in [8]. An alternate deterministic equivalent formulation uses coupling first-stage variables that are common for all scenario blocks, yielding a primal decomposable problem that can be solved by schemes such as Benders' when all second-stage variables are continuous. Extensions have been proposed, but these tend to be computationally impractical, especially when both stages have continuous and integer variables [9].

Progressive Hedging (PH) was designed specifically for stochastic programming problems and combines the idea of augmented Lagrangian methods with a scenario-based decomposition. Unlike branch and price and other approaches based on column generation, the subproblems in PH are updated without needing to iteratively solve a master problem.

B. The Progressive Hedging Algorithm

For convex optimization problems, such as stochastic linear programs, PH is guaranteed to converge to the optimal solution [10], even if the subproblems are solved approximately [11]. In our case, since both first and second stage decisions contain integer variables, theoretical convergence is lost. Nevertheless,

PH can be applied to non-convex stochastic integer programs (such as ours) to obtain heuristic solutions [12].

In PH, for iteration i , a subproblem is defined for each scenario s as:

$$(SP_s^i) \quad \min p_s (c^T x_s + q_s^T y_s) + \hat{f}_s^i(x_s, i) \quad (2a)$$

$$Ax_s = b \quad (2b)$$

$$T_s x_s + W_s y_s = h_s \quad (2c)$$

$$x \in X, y \in Y \quad (2d)$$

Denoting the optimal solution to subproblem s in iteration i as x_s^{*i} , the *penalty function* $\hat{f}_s^i(x_s, i)$ is defined as:

$$\hat{f}_s^i(x_s, i) = \lambda_s^i x_s + \frac{1}{2} \rho^i (x_s - \bar{x}^i)^2 \quad \forall i > 0 \quad (3)$$

$$\hat{f}_s^i(x_s, i) = 0 \quad i = 0 \quad (4)$$

where $\rho^i > 0$ for all i is the “*penalty factor*,” \bar{x} is a vector defined by $\bar{x}^i = \sum_{s \in S} p_s x_s^{*i-1}$, and λ_s^i is defined as

$$\lambda_s^i = \lambda_s^{i-1} + \rho^{i-1} (x_s^{*i-1} - \bar{x}^i) \quad \forall i > 0 \quad (5)$$

$$\lambda_s^i = 0 \quad i = 0 \quad (6)$$

At each iteration of the algorithm, problem SP_s^i is solved for every scenario, and the optimal solutions x_s^* are used to update the penalty function. The goal of the penalty function is to ultimately guide all first-stage variables x^* to satisfy constraint (1a), giving a feasible solution.

PH halts execution when convergence is reached for all the first-stage variables. In a commonly used termination criterion [13], PH terminates when the norm

$$\delta = \{ \|\bar{x}^i - \bar{x}^{i-1}\|^2 + \sum_{s \in S} p_s \|x_s^{*i} - \bar{x}^i\|^2 \}^{\frac{1}{2}}$$

drops below a certain parametric threshold. In this case δ is a measure of the “distance from convergence”. As we discuss later, we also consider termination criteria based on the gap between the best-known upper bound (from feasible solutions) and the best-known lower bound (obtained by combining the solutions of the subproblems for the first iteration). In practice, many problems also consider a global execution time limit.

The main drawbacks of PH are that the common penalty function produces a quadratic integer program for each subproblem, which can present computational challenges; furthermore, the PH scheme does not provide feasible solutions to

the stochastic problem until it converges (or at least all the integer-first stage variables converge). Even then, there is no guarantee that a feasible solution exists for the original problem with fixed first-stage integer variables. We address both these shortcomings in mPH, our modified PH-based algorithm for solving two-stage stochastic optimization problems.

III. MPH: A PH-BASED HEURISTIC FOR TWO-STAGE STOCHASTIC PROBLEMS

We motivate our introduction of mPH, a modified version of PH effective in solving large real-world stochastic problems, by detailing our approach in overcoming the drawbacks of PH.

A. Linear subproblems

The penalty function defined in (3) and (4) leads to quadratic subproblems. If all the second-stage variables are defined as binaries there is an equivalent linear formulation for SP_s^i [14], but this does not apply in our unit commitment problems. When the size of the problem increases, the quadratic subproblems quickly become difficult to solve if not intractable. In theory, the presence of the quadratic term ensures convergence to an optimal solution for the convex case (as in stochastic linear programs). From our perspective, losing the convergence property by modifying the penalty function is acceptable since the proof of convergence does not extend to the case of non-convex stochastic integer programs.

As illustrated in Figure 2(a), the quadratic term in $\hat{f}_s'(x_s, i)$ pushes the first-stage variables to assume values that are near the average \bar{x}^i . In a linear penalty function obtained by removing the quadratic term, the deviation from \bar{x}^i is not penalized, causing oscillatory behavior of the variables and impacting convergence of the algorithm. Therefore, instead of removing the quadratic term, we approximate the quadratic distance from \bar{x}^i with the absolute distance, as in Figure 2(b). To the best of our knowledge, this is a new technique in the context of PH. For a generic scenario s , we define the following penalty function:

$$\hat{f}_s(x_s, i) = \lambda_s^i x_s + \frac{1}{2} \rho^i |x_s - \bar{x}^i| \quad (7)$$

The intent behind the definition of $\hat{f}_s(x_s, i)$ is to mitigate the oscillatory behaviour and allow the solution of linear subproblems at the same time. Note that with the linear penalty the overall cost of a given first-stage variable $x_{s,j}$ has a minimum at \bar{x}_j^i if penalty factor $\rho_j^i \geq 2(c_j + \lambda_{s,j}^i)$, or at the minimum defined by c and λ_s^i , otherwise (as in Figure 2(c)).

Choosing the correct penalty factor for the original quadratic penalty function is critical and is often data dependent ([12], [15]). When $\hat{f}_s(x_s, i)$ is used instead of $\hat{f}_s'(x_s, i)$, the smoothness of the quadratic penalization is lost and choosing the correct penalty factor is as critical as in the quadratic case.

Penalty factor: We extend the element-specific penalty factor first proposed in [12]. The authors define a penalty factor, for each iteration i and each first-stage variable j , as follows:

$$\tilde{\rho}_j^i = \frac{|c_j|}{(\max_s x_{s,j}^{*i} - \min_s x_{s,j}^{*i})}.$$

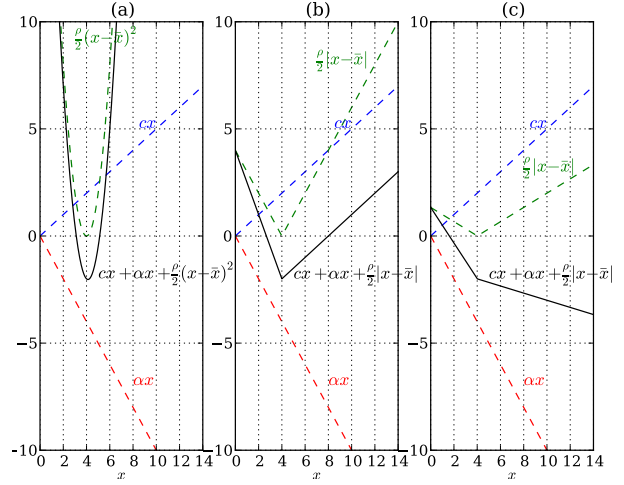


Figure 2. Examples of penalty functions: for a given first-stage variable with original cost $c = 0.5$, in (a) the penalty function $\hat{f}_s'(x_s, i)$ is represented supposing $\lambda_s^i = -1$, $\rho^i = 2$; in (b) and (c), the penalty function $\hat{f}_s(x_s, i)$ is represented with same λ_s^i and, respectively, $\rho^i = 2$ and $\rho^i = \frac{2}{3}$.

With absolute distances, using the same definition results in a weak penalty factor and, consequently, slow convergence of the algorithm. Instead, we use the vector $\rho^i = (\rho_1^i, \dots, \rho_n^i)$, where $\rho_j^i = (\tilde{\rho}_j^i)^2$ and n is the number of first-stage variables.

In $\hat{f}_s'(x_s, i)$, any $\rho > 0$ guarantees that the subproblems are bounded. This is not the case when we consider absolute distances. To overcome this, for every unbounded first-stage variable j , we adjust $\tilde{\rho}_j^i$ so that the overall cost defined by c , λ and $\tilde{\rho}_j^i$ is greater than zero for any $x_{s,j} \geq \bar{x}_j^i$. In particular if we observe for a positive unbounded variable j that:

$$\tilde{\rho}_j^i \leq -2 \min_{s: (c_j^s + \lambda_j^{s,v}) < 0} (c_j^s + \lambda_j^{s,v})$$

we impose:

$$\tilde{\rho}_j^i = -2 \min_{s: (c_j^s + \lambda_j^{s,v}) < 0} (c_j^s + \lambda_j^{s,v}) + \gamma$$

where $\gamma > 0$. In our experiments, we set $\gamma = 1$.

B. Guided MIP Solves for feasible solutions

PH does not directly provide feasible solutions until complete convergence is attained. For difficult to solve instances, it is highly likely that convergence will not occur within a given time limit. Existing strategies typically stop the algorithm before convergence is reached for all the first stage variables and then search for global feasible solutions by fixing some variables. Our new approach searches for feasible solutions while the algorithm iterates. Our ability to find good feasible solutions even after the first iteration is particularly useful for instances such as ours with large and difficult subproblems, in which a single iteration of mPH requires a considerable computational effort. Moreover, different iterations of the algorithm may generate different feasible solutions by exploring different parts of the feasibility region.

Guided solves: We refer to first-stage variables that have the same value in the solutions of all subproblems for a given iteration as “agreed” variables. We obtain feasible solutions by optimizing SUC with agreed variables fixed to the common value. We call this problem SUC' and its solution as a

Instance	gap % mPH		Instance	gap % mPH	
	It 1	It 2		It. 1	It. 2
D2020-03-02	0.1	0.11	D2020-12-29	0.01	0.02
D2020-01-04	0.13	0.11	D2020-10-10	0.01	0.01
D2020-05-14	0.09	0.09	D2020-07-16	0.87	-
D2020-08-07	-	0.13	D2020-06-30	0.23	0.12
D2020-09-27	0.03	0.02	D2020-04-15	0.02	0.02
D2020-11-09	0.05	0.06	D2020-02-12	0.00	0.00

Table I
COMPARISON OF SOLUTION QUALITY FOR 5-SCENARIOS INSTANCES

“guided solve.” Since we may have different sets of agreed variables in different iterations, we may obtain a different SUC' formulation at every iteration, producing a new solution.

For continuous first-stage variables, we define a tolerance in which two first-stage variables can be considered equal and therefore agreed. For any $\epsilon > 0$, we denote by “ ϵ -agreed” all the variables for which the variance from a common value is less than ϵ . All the ϵ -agreed variables can then be fixed at their average value \bar{x}^i . If ϵ is smaller than the feasibility tolerance of the MIP solver used for the guided solves, then the definition of agreed and ϵ -agreed variables coincide.

SUC' feasibility: Observe that Problem SUC' is obtained by adding constraints to the original SUC formulation; thus the feasibility region of SUC' is contained in the region of SUC . On the other hand, there is no guarantee that SUC' has feasible solutions even if SUC does. In fact, it is not always possible to satisfy the non-anticipativity constraints for all non-fixed first-stage variables for all agreed variables. However, if a combination of first-stage variables is known to be feasible for all scenarios if it is feasible for one scenario, then is possible to state a-priori that SUC' is always feasible if SUC is feasible.

ϵ -strategies: The parametric definition of ϵ -agreed variables suggests certain natural strategies. When the number of agreed first-stage variables is low, the guided solve requires a computational effort similar to the original SUC formulation. In this case, considering integer variables that have similar (unequal) values as ϵ -agreed may help. This causes an increased number of fixed variables in SUC' , with consequent reduction of required computation and, potentially, solution quality.

Conversely, one may want to reduce the number of fixed first-stage variables, when easy to solve SUC' problems return poor or infeasible solutions. One strategy is to fix variables that converged only in the previous iteration. This idea arises from the fact that some generation resources are used only when uncertainty is considered. For example, the subproblems solved separately may not use such resources in the initial iterations, even if they are used in the optimal solution. In this case, fixing such resources to zero will cut off the optimum.

IV. COMPUTATIONAL RESULTS

Instances and Experimental Setup: We evaluate our mPH algorithm on a testbed containing twelve 5-scenario instances and one 20-scenario instance. Each instance corresponds to an hour-timestep unit commitment problem for a single day in the simulated year 2020, as described in Section I-A. We ran our mPH algorithm on a Power7 processor, with 8 cores running at

Instance	Times [s]		
	Direct 0.05-Opt	mPH Serial	mPH Parallel
D2020-03-02	9923.30	1642	463.3
D2020-01-04	5318.20	2503.1	643.9
D2020-05-14	13708.30	1668.3	513.1
D2020-08-07	8097.70	1269.5	412.6
D2020-09-27	11842.70	1381.4	491.7
D2020-11-09	5484.30	1543.6	380.6
D2020-12-29	9192.00	2699	691.7
D2020-10-10	35951.50	3923.7	1020.7
D2020-07-16	2965.80	1221.5	301
D2020-06-30	5988.80	1110.9	321.1
D2020-04-15	36022.50	3650.4	1151.3
D2020-02-12	7642.10	1748.2	483.8
AVERAGES	12678.1	2030.13	572.90

Table II
COMPARISON OF EXECUTION TIMES FOR 5-SCENARIO INSTANCES

3.61GHz, each capable of four-way simultaneous multithreading. Power7 executes instructions out-of-order. There are 12 execution units (including 2 fixed-point units and 2 load/store units) per core shared by the 4 hardware threads.

A. Results

We compare mPH with “direct” solutions of the SUC formulation using CPLEX 12.4 with standard parameters except for “relative MIP gap tolerance” which is set to 0.05%. We use the same version of CPLEX and the same parameter settings to solve SUC' in mPH, but we set the relative MIP gap tolerance to 0.5% for the subproblems SP_s^i .

To reduce the computational requirements for solving SUC' , we fixed some second-stage variables in addition to the first-stage “ ϵ -agreed” variables. Any second-stage variable that assumes the same value in the optimal solutions of all the subproblems is fixed to this agreed value in SUC' .

Five scenarios instance: In Table I we summarize the quality of the results achieved for the 5-scenarios instances. In column “It. 1” and “It. 2” we report the gap, in percentage, between the optimal solutions obtained using the direct approach and the feasible solutions returned by the guided solves after the first two iterations. In our experiments, we stopped mPH after two iterations since we obtained solutions comparable to the optimum in just two iterations. We did obtain infeasible SUC' problems in two cases, but the overall performance was not compromised since for every instance it was possible to reach a feasible solution in either the first or second iteration.

Table II compares the solution times (in seconds) for the 5-scenario problems. We achieve a 6 times speedup in the average serial solution time. We also tested a parallel implementation that solves all the subproblems in parallel, under which the guided solve subsequent to iteration i is also carried out in parallel with the subproblems of $i + 1$. In this implementation, we find a 22 times average speedup.

In our experiments, we observed a decrease in subproblem solution times for the second iteration, which is partially due to information from the first iteration and partially due to a different (penalizing) objective function. We also observe a reduction in solution time for the guided solves, averaging a 40% improvement. In our experiments, the guided solve re-optimized from scratch at every iteration, with no information

It.	SP_s^i Sum	Times [s]		Gap % SUC'_i -LB
		SP_s^i Max	SUC'_i	
1	7172.88	466.37	8781.04	0.02
2	2600.46	186.71	2419.81	0.02
3	3012.23	214.35	2072.5	0.01
4	2543.42	202.39	2277.18	0.01
5	2786.57	260.27	1732.35	0.01
6	3614.93	439.67	2120.29	0.01
7	4033.06	382.32	1218.11	0.02

Table III
RESULTS FOR THE 20-SCENARIOS INSTANCE

used from the previous iteration, so the reduced solution times can only be attributed to the different fixed variables in SUC' .

Twenty scenarios instance: Table III lists mPH execution times and solution quality for the 20-scenario instance. Time columns are the same as in the 5-scenario case, and the rows represent the iteration number. The gap between the solutions from the guided solves and the best known lower bound is reported in column “Gap % SUC'_i -LB”. It is clear from the results that the solutions provided by the guided solves are extremely close to optimal. The feasible solutions returned by mPH are the only valid upper bound in this case as the 20-scenarios instance is intractable when solved with the direct approach (no feasible solution was found in 24 hours).

Convergence: While mPH provided near optimal solutions after 2 iterations, the convergence of the algorithm depends on the effectiveness of the linear penalty function $\hat{f}_s(x_s, i)$. To gauge this, we ran the algorithm for ten iterations for the 5-scenarios instances and seven iterations for the 20-scenario instance; there were no cases of complete convergence.

In Figure 3 we report the values of δ . After the first iteration the value of δ in the 5-scenarios instances becomes very small and remains there. In the 20-scenario instance, we observed a slight but constant reduction of δ in the following iterations.

To achieve complete algorithmic convergence, one could fix the agreed variables both in SUC' and also in the subproblems of following iterations. Alternatively, one could use a definition of ϵ^i that increases when the iterations of i increases.

V. CONCLUSIONS

Our mPH algorithm produces near-optimal solutions for our stochastic unit commitment problems, with a 6 times improvement in serial solution time over the standard approach, and a 22 times improvement in parallel. We can now solve

stochastic problems with many more renewable scenarios than previously achievable, including the solution of the formerly intractable 20-scenario case. Given the recent interest in the power industry for stochastic unit commitment models as a way of managing uncertainty [16], these results represent a solid tool that can be used to mitigate the computational burden of such problems.

The methods proposed here suggest several areas for further analysis. First, as mentioned in the ϵ -strategies paragraph, there are different possible definitions of the agreed variables. Second, when feasible solutions are available there is the possibility (not exploited in this work) to use these solutions to help convergence of PH or speed up the resolution process in general. For example, if in a given iteration SUC' returns a particularly good feasible solution, it may be advantageous to fix some of the agreed variables in the subproblems. This will help produce easier subproblems, with a consequent reduction of computation, and may also help convergence of the algorithm by progressively fixing more variables.

REFERENCES

- [1] California State Senate, “Bill Number 2,” April 12 2011.
- [2] T. Edmunds, A. Lamont, V. Bulaevskaya, C. Meyers, J. Mirocha, A. Schmidt, M. Simpson, S. Smith, P. Sotorrio, P. Top, and Y. Yao, “The value of storage and demand response for renewable integration,” California Energy Commission, Tech. Rep. under contract CEC-500-10-051, 2013, awaiting approval for public release.
- [3] California Independent System Operator, “Integration of renewable resources: Technical appendices for California ISO renewable integration studies,” California ISO, Tech. Rep., 2010.
- [4] Energy Exemplar, “PLEXOS Integrated Energy Modeling Software,” 2013, <http://www.energyexemplar.com/>.
- [5] C. C. Carøe and R. Schultz, “Dual decomposition in stochastic integer programming,” *Ops Research Letters*, vol. 24, pp. 37–45, 1999.
- [6] A. Belloni, A. D. S. Lima, M. P. Maceira, and C. A. Sagastizábal, “Bundle relaxation and primal recovery in unit commitment problems, the Brazilian case,” *Annals of Ops Research*, vol. 120, pp. 21–44, 2003.
- [7] G. Lulli and S. Sen, “A branch-and-price algorithm for multistage stochastic integer programming with application to stochastic batch-sizing problems,” *Management Science*, vol. 50, pp. 786–796, 2004.
- [8] R. T. Rockafellar and R. J.-B. Wets, “Scenarios and policy aggregation in optimization under uncertainty,” *Mathematics of operations research*, vol. 16, pp. 119–147, 1991.
- [9] D. Gade, S. Küçükyavuz, and S. Sen, “Decomposition algorithms with parametric gomory cuts for two-stage stochastic integer programs,” *Mathematical Programming*, pp. 1–26, 2012.
- [10] T. Helgason and S. W. Wallace, “Approximate scenario solutions in the progressive hedging algorithm,” *Annals of Ops Research*, vol. 31, pp. 425–444, 1991.
- [11] R. J. Wets, “The aggregation principle in scenario analysis and stochastic optimization,” in *Algorithms and model formulations in mathematical programming*. Springer, 1989, pp. 91–113.
- [12] J.-P. Watson, D. L. Woodruff, and D. R. Strip, “Progressive hedging innovations for a stochastic spare parts support enterprise problem,” *Naval Research Logistics*, 2007.
- [13] A. De Silva and D. Abramson, “Computational experience with the parallel progressive hedging algorithm for stochastic linear programs,” in *Proceedings of 1993 Parallel Computing and Transputers Conference Brisbane*. Citeseer, 1993, pp. 164–174.
- [14] T. G. Crainic, X. Fu, M. Gendreau, W. Rei, and S. W. Wallace, “Progressive hedging-based metaheuristics for stochastic network design,” *Networks*, vol. 58, pp. 114–124, 2011.
- [15] J. M. Mulvey and H. Vladimirou, “Solving multistage stochastic networks: An application of scenario aggregation,” *Networks*, vol. 21, pp. 619–643, 1991.
- [16] P. Ruiz, C. Philbrick, E. Zak, K. Cheung, and P. Sauer, “Uncertainty management in the unit commitment problem,” *IEEE Transactions on Power Systems*, vol. 24, pp. 642–651, 2009.

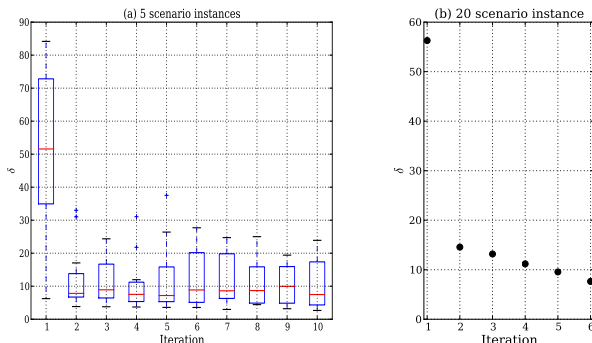


Figure 3. Evolution of δ .